ICTC - Khóa BASIC IC DESIGN

BÀI TẬP LỚN MÔN IC -DESIGN

GVHD: Anh Trần Hồng Ân

LAB1:**TIMER-IP**

STT	Họ và tên	MSSV
1	Lê Duy Thức	2112416

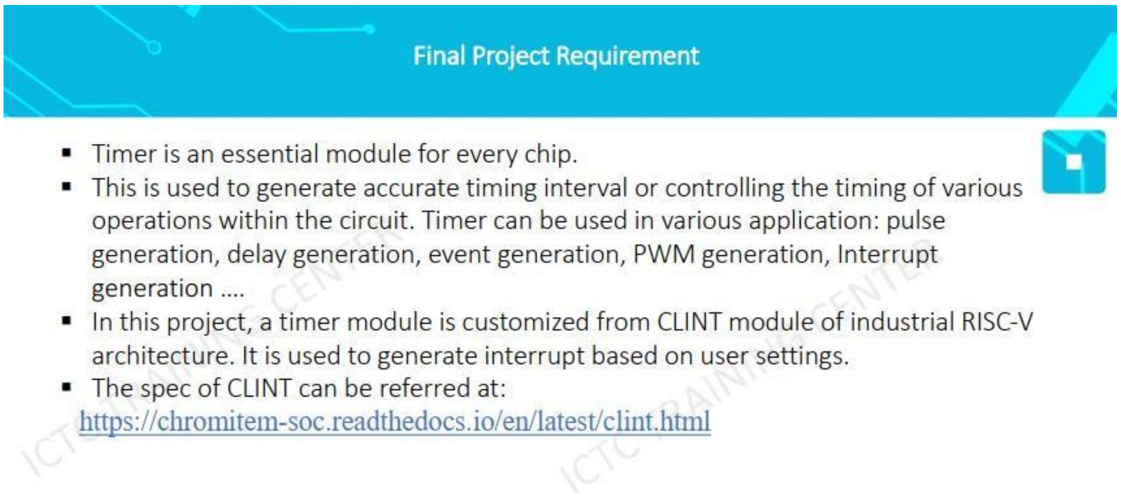
WORK DIRECTORY : /ictc/student-data/duythuc/FINAL_project/final

Lớp: IC8 Số thứ tự nhóm: N/A

STT	Họ và tên	MSSV	Nhiệm vụ	Hoàn thành
1	Lê Duy Thức	2112416	Spec, check_list, RTL, testbench coverage testcase, debug	100%

SPECIFICATION

1. Overview


Final Project Requirement

- Timer is an essential module for every chip.
- This is used to generate accurate timing interval or controlling the timing of various operations within the circuit. Timer can be used in various application: pulse generation, delay generation, event generation, PWM generation, Interrupt generation ....
- In this project, a timer module is customized from CLINT module of industrial RISC-V architecture. It is used to generate interrupt based on user settings.
- The spec of CLINT can be referred at:

https://chromitem-soc.readthedocs.io/en/latest/clint.html

2. Functional Requirements

2.1 General Features

- 64-bit count-up timer.
- APB Slave interface for register configuration. 32bit trans
- Address space: 4KB (0x4000_1000 – 0x4000_1FFF).

- System clock frequency: 200 MHz.
- Active low asynchronous reset (`sys_rst_n`).

2.2 Counting Modes

- **Default mode:** The counter increments based on the system clock.
- **Control mode:** Counter speed can be adjusted using a divisor value (`div_val`) when the control mode is enabled (`div_en`).

(TCR.div_en ==1) - TCR [1]
(TCR.div_val == n) - TCR [11:8]

- **Halted mode**: The timer can be halted in debug mode. The halt is triggered when `debug_mode` is high and a halt request (`THCSR.halt_req`) is asserted. The timer resumes when the halt request is cleared.

(Debug_mode ==1)

(THCSR.halt_req ==1) - THCSR [0]

- **Error handling mode**: In control mode, changes to the divisor settings (`div_val`) while the timer is running results in an error response.

(TCR.div_val changed while running)
(TCR.div_en changed while running)
(TCR.div_val loaded with invalid input)

- **Byte access**: The bus supports byte access to the individual registers. ????
- **Wait states**: A 1-cycle wait state can be inserted during APB transactions to improve timing.

2.3 Timer Interrupt

- Interrupt generation: An interrupt (`tim_int`) is triggered when the counter matches the compare value (`TCMP0` and `TCMP1`).
- Interrupt can be enabled or disabled via the `TIER` register.
(TIER [0]==1)
- Interrupt status can be cleared by writing to the `TISR.int_st` bit.
(TISR [0]==1)

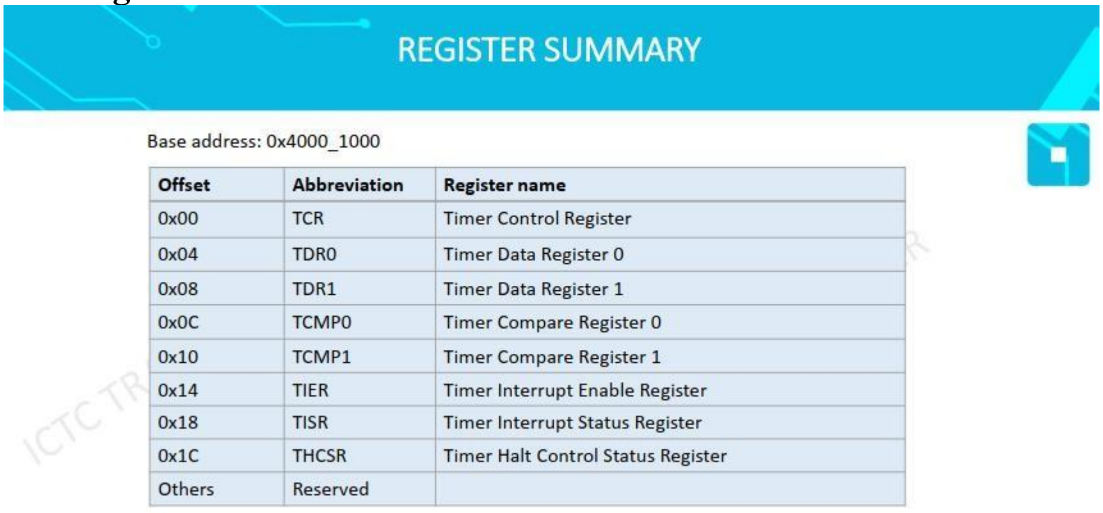
3. Design Details

3.1 APB Slave Interface

The Timer IP communicates with an external master device via the APB interface. The APB interface includes the following signals:

- tim_psel: Select signal for the timer.
- tim_pwrite: Write enable signal.
- tim_penable: APB enable signal.
- tim_paddr: 32-bit address bus.
- tim_pwdata: 32-bit write data bus.
- tim_prdata: 32-bit read data bus.
- tim_pready: Ready signal indicating the transfer is complete.
- tim_pslverr: Error response signal.

3.2 Register Set

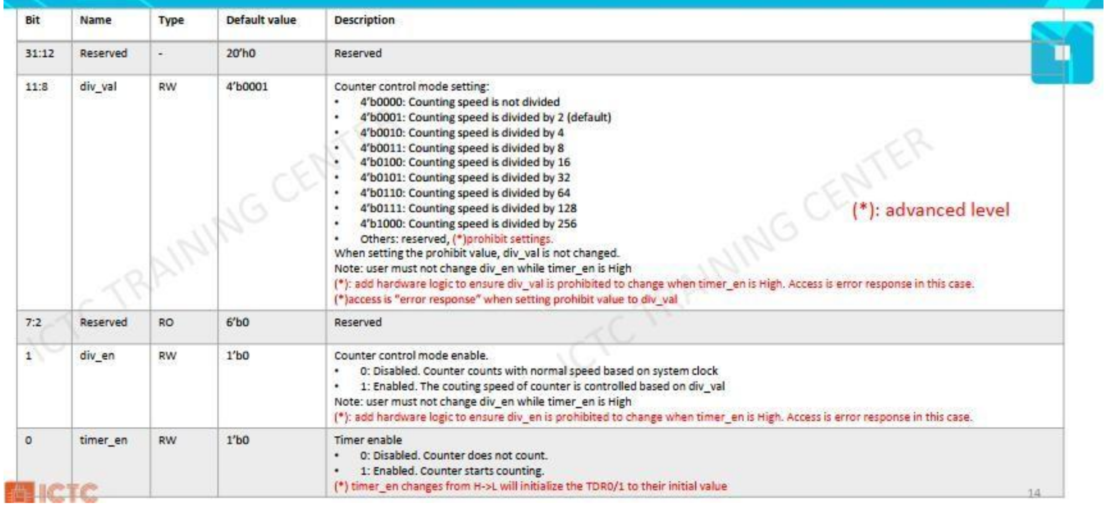


REGISTER SUMMARY

Base address: 0x4000_1000

Offset	Abbreviation	Register name
0x00	TCR	Timer Control Register
0x04	TDR0	Timer Data Register 0
0x08	TDR1	Timer Data Register 1
0x0C	TCMP0	Timer Compare Register 0
0x10	TCMP1	Timer Compare Register 1
0x14	TIER	Timer Interrupt Enable Register
0x18	TISR	Timer Interrupt Status Register
0x1C	THCSR	Timer Halt Control Status Register
Others	Reserved	

3.3 Timer Control Register (TCR)



Bit	Name	Type	Default value	Description
31:12	Reserved	-	20'h0	Reserved
11:8	div_val	RW	4'b0001	Counter control mode setting: • 4'b0000: Counting speed is not divided • 4'b0001: Counting speed is divided by 2 (default) • 4'b0010: Counting speed is divided by 4 • 4'b0011: Counting speed is divided by 8 • 4'b0100: Counting speed is divided by 16 • 4'b0101: Counting speed is divided by 32 • 4'b0110: Counting speed is divided by 64 • 4'b0111: Counting speed is divided by 128 • 4'b1000: Counting speed is divided by 256 • Others: reserved, (*)prohibit settings. When setting the prohibit value, div_val is not changed. Note: user must not change div_en while timer_en is High (*): add hardware logic to ensure div_val is prohibited to change when timer_en is High. Access is error response in this case. (*)access is "error response" when setting prohibit value to div_val
7:2	Reserved	RO	6'b0	Reserved
1	div_en	RW	1'b0	Counter control mode enable. • 0: Disabled. Counter counts with normal speed based on system clock • 1: Enabled. The couting speed of counter is controlled based on div_val Note: user must not change div_en while timer_en is High (*): add hardware logic to ensure div_en is prohibited to change when timer_en is High. Access is error response in this case.
0	timer_en	RW	1'b0	Timer enable • 0: Disabled. Counter does not count. • 1: Enabled. Counter starts counting. (*) timer_en changes from H->L will initialize the TDR0/1 to their initial value

(*): advanced level

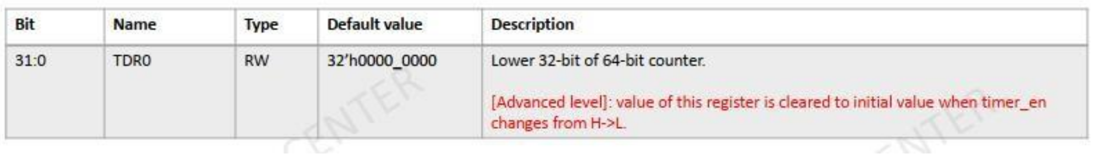
ICTC

14

3.4 Timer Data Registers (TDR0/TDR1)

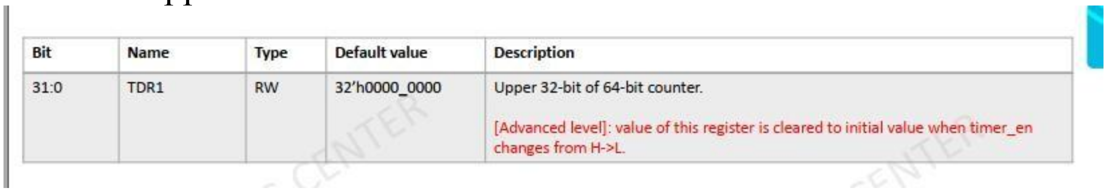
These registers store the 64-bit counter value.

- TDR0: Lower 32 bits of the 64-bit counter.



Bit	Name	Type	Default value	Description
31:0	TDR0	RW	32'h0000_0000	Lower 32-bit of 64-bit counter. [Advanced level]: value of this register is cleared to initial value when timer_en changes from H->L.

- TDR1: Upper 32 bits of the 64-bit counter.

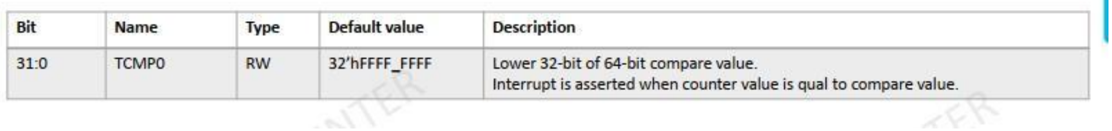


Bit	Name	Type	Default value	Description
31:0	TDR1	RW	32'h0000_0000	Upper 32-bit of 64-bit counter. [Advanced level]: value of this register is cleared to initial value when timer_en changes from H->L.

3.5 Timer Compare Registers (TCMP0/TCMP1)

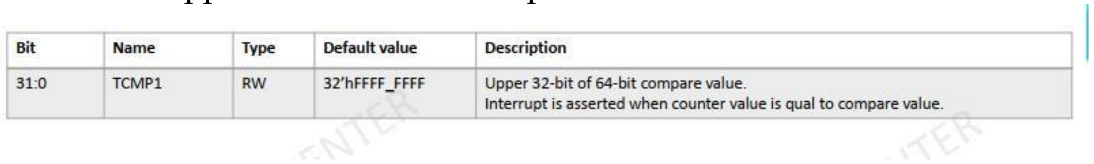
These registers hold the 64-bit compare value. When the counter reaches this value, an interrupt is generated.

- TCMP0: Lower 32 bits of the compare value.



Bit	Name	Type	Default value	Description
31:0	TCMP0	RW	32'hFFFF_FFFF	Lower 32-bit of 64-bit compare value. Interrupt is asserted when counter value is qual to compare value.

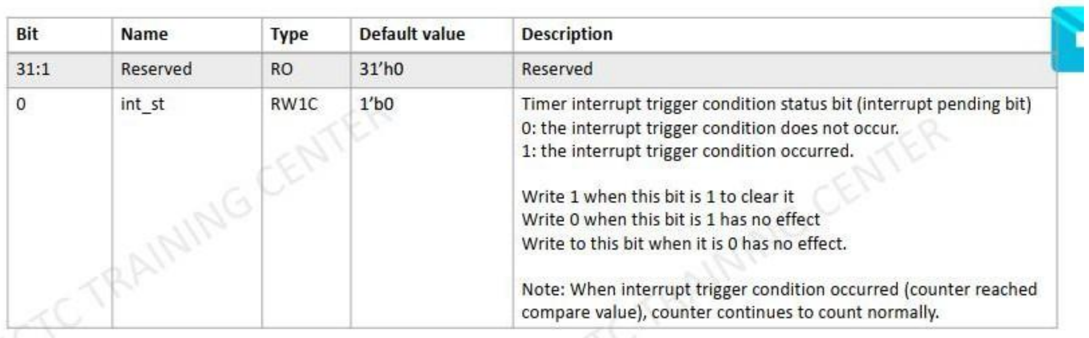
- TCMP1: Upper 32 bits of the compare value.



Bit	Name	Type	Default value	Description
31:0	TCMP1	RW	32'hFFFF_FFFF	Upper 32-bit of 64-bit compare value. Interrupt is asserted when counter value is qual to compare value.

3.6 Timer Interrupt Enable Register (TIER)

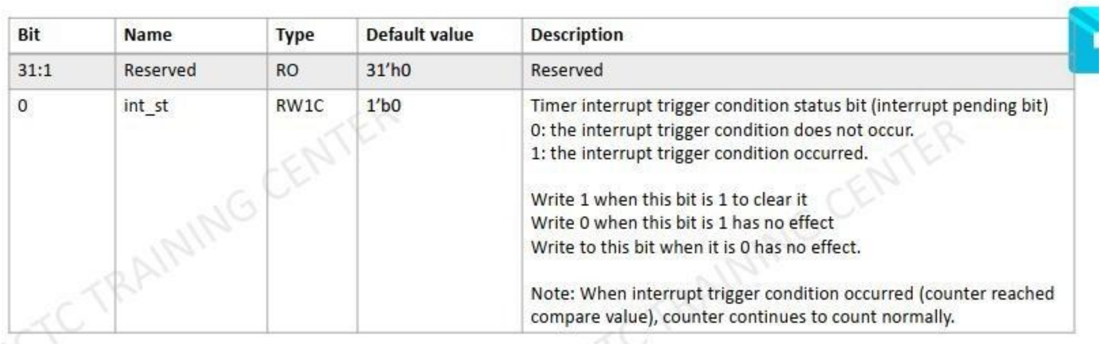
This register controls whether the timer interrupt is enabled.



Bit	Name	Type	Default value	Description
31:1	Reserved	RO	31'h0	Reserved
0	int_st	RW1C	1'b0	Timer interrupt trigger condition status bit (interrupt pending bit) 0: the interrupt trigger condition does not occur. 1: the interrupt trigger condition occurred. Write 1 when this bit is 1 to clear it Write 0 when this bit is 1 has no effect Write to this bit when it is 0 has no effect. Note: When interrupt trigger condition occurred (counter reached compare value), counter continues to count normally.

3.7 Timer Interrupt Status Register (TISR)

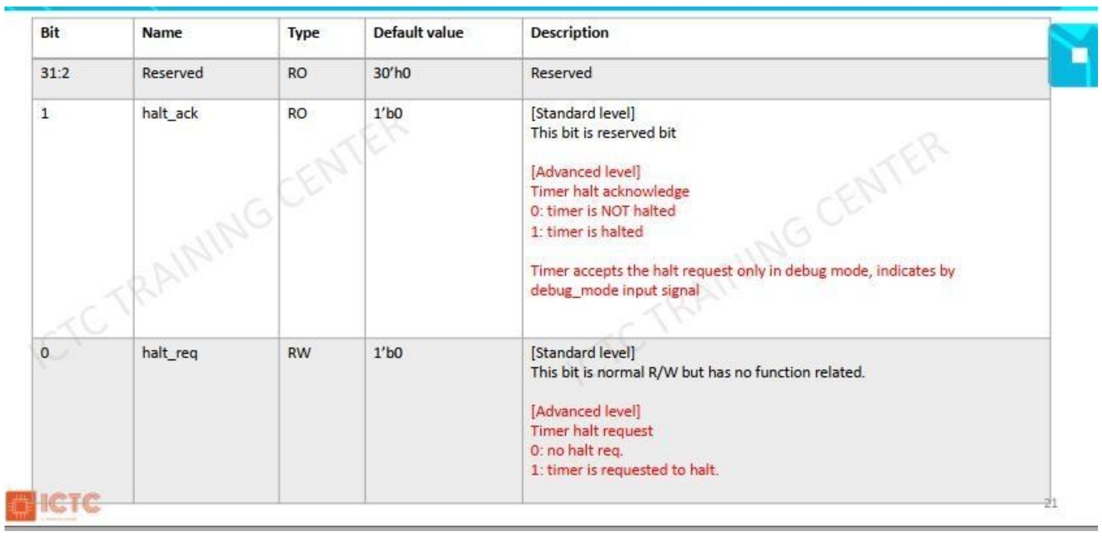
This register indicates whether an interrupt has occurred. It must be cleared by software after the interrupt is handled.



Bit	Name	Type	Default value	Description
31:1	Reserved	RO	31'h0	Reserved
0	int_st	RW1C	1'b0	Timer interrupt trigger condition status bit (interrupt pending bit) 0: the interrupt trigger condition does not occur. 1: the interrupt trigger condition occurred. Write 1 when this bit is 1 to clear it Write 0 when this bit is 1 has no effect Write to this bit when it is 0 has no effect. Note: When interrupt trigger condition occurred (counter reached compare value), counter continues to count normally.

3.8 Timer Halt Control Status Register (THCSR)

This register controls the halt mode operation of the timer.



Bit	Name	Type	Default value	Description
31:2	Reserved	RO	30'h0	Reserved
1	halt_ack	RO	1'b0	[Standard level] This bit is reserved bit [Advanced level] Timer halt acknowledge 0: timer is NOT halted 1: timer is halted Timer accepts the halt request only in debug mode, indicates by debug_mode input signal
0	halt_req	RW	1'b0	[Standard level] This bit is normal R/W but has no function related. [Advanced level] Timer halt request 0: no halt req. 1: timer is requested to halt.

4. Signal List

Top module : timer_top

Signal name	**Width**	**Direction**	**Description**	**Type**
Sys_clk	1	input	System clk	wire
Sys_rst_n	1	input	Reset sig	wire
Tim_psel	1	input	APB select signal for time	wire
tim_pwrite	1	input	APB write enable sig	wire
Tim_penable	1	input	APB enable sig	wire
Tim_addr	32	input	APB address bus for reg select	wire
Tim_pwdata	32	input	APB write data bus	wire
Tim_prdata	32	output	APB read data bus	wire
Tim_pready	1	output	APB ready sig	wire
Tim_pslverr	1	output	APB error sig	wire
Tim_int	1	output	Time interrupt sig	wire
Dbg_mode	1	output	Debug dev mode	wire

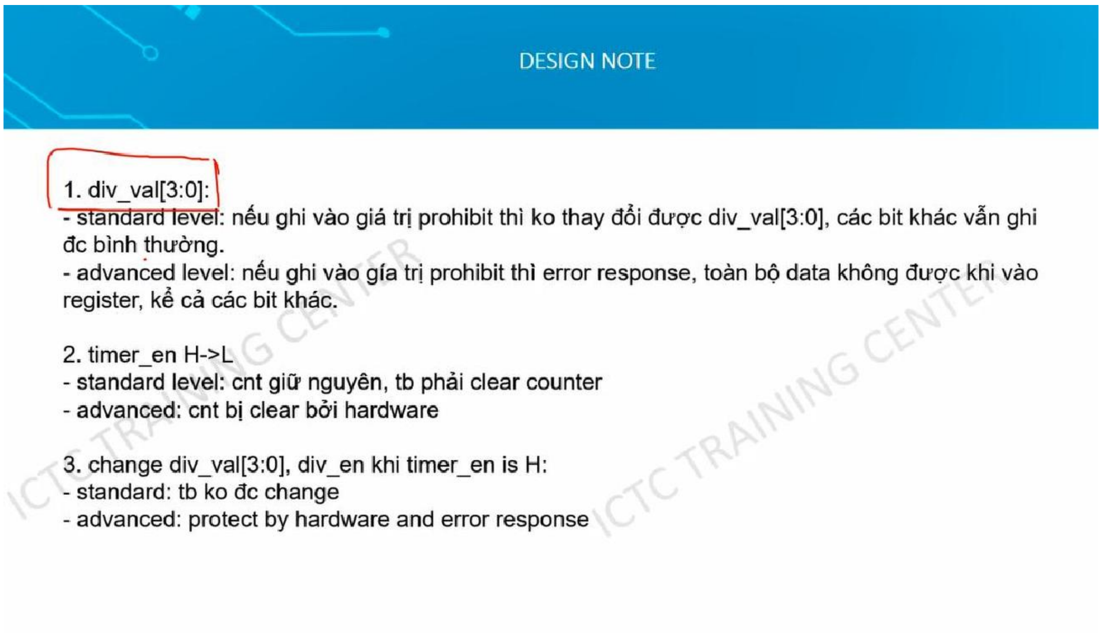
Module timer_ctrl

Signal name	**Width**	**Direction**	**Description**	**Type**
Sys_clk	1	input	System clk	wire
Sys_rst_n	1	input	Reset sig	wire
Timer_en	1	input	Timer enable sig	wire
Div_en	1	input	Division enable sig	wire
Div_val	4	input	Division val sig	wire
Tcmp	64	input	Timer data reg	wire
Halt_req	1	input	Halt sig	wire
Int_en	1	input	Interrupt enable sig	wire
Dbg_mode	1	input	Dev mode enable sig	wire
tdr	64	output	Timer data reg	reg
Halt_ack	1	output	Halt acknowledge sig	reg
Int_st	1	output	Interrupt status sig	reg
Tim_int	1	output	Timer interrupt sig	reg

Module APB

Signal name	Width	Direction	Description	Type
Sys_clk	1	input	System clk	wire
Sys_rst_n	1	input	Reset sig	wire
Tim_psel	1	input	Timer enable sig	wire
Tim_pwrite	1	input		wire
Tim_penable	1	input		wire
Tim_paddr	32	input		wire
Tim_pwdata	32	input		wire
Tim_prdata	32	input		wire
Halt_ack	1	input		wire
Int_st	1	input		wire
Tim_prdata	32	output		reg
Tim_pready	1	output		reg
Tim_pslverr	1	output		reg
Timer_en	1	output		reg
Div_en	1	output		reg
Div_val	4	output		reg
Tdr	64	output		reg
Tcmp	64	output		reg
Halt_req	1	output		reg
Int_en	1	output		reg

Design note

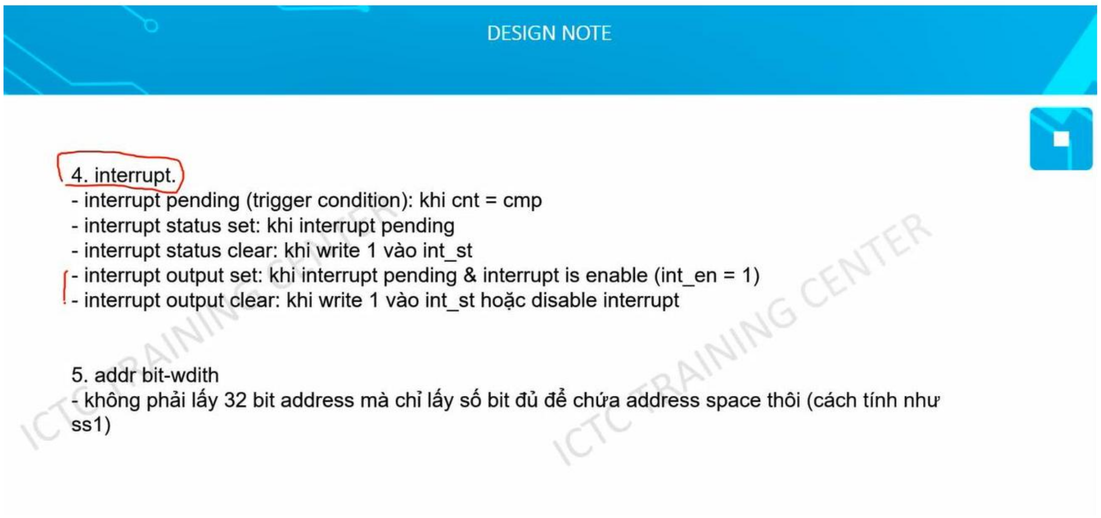


DESIGN NOTE

1. div_val[3:0]:
- standard level: nếu ghi vào giá trị prohibit thì ko thay đổi được div_val[3:0], các bit khác vẫn ghi đc bình thường.
- advanced level: nếu ghi vào gía trị prohibit thì error response, toàn bộ data không được khi vào register, kể cả các bit khác.

2. timer_en H->L
- standard level: cnt giữ nguyên, tb phải clear counter
- advanced: cnt bị clear bởi hardware

3. change div_val[3:0], div_en khi timer_en is H:
- standard: tb ko đc change
- advanced: protect by hardware and error response

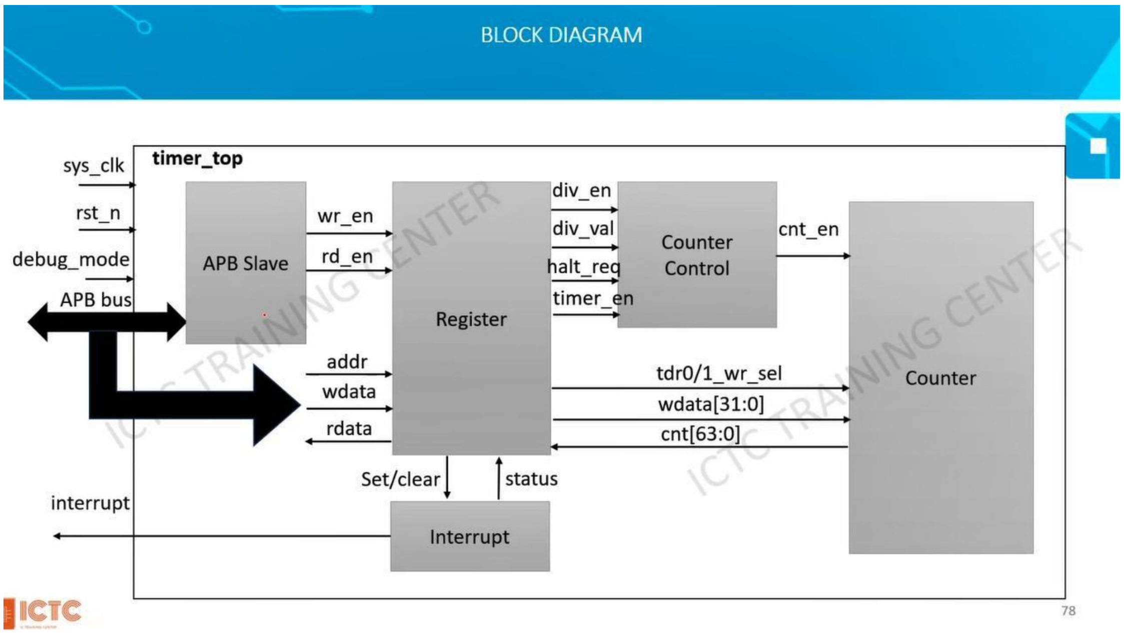


DESIGN NOTE

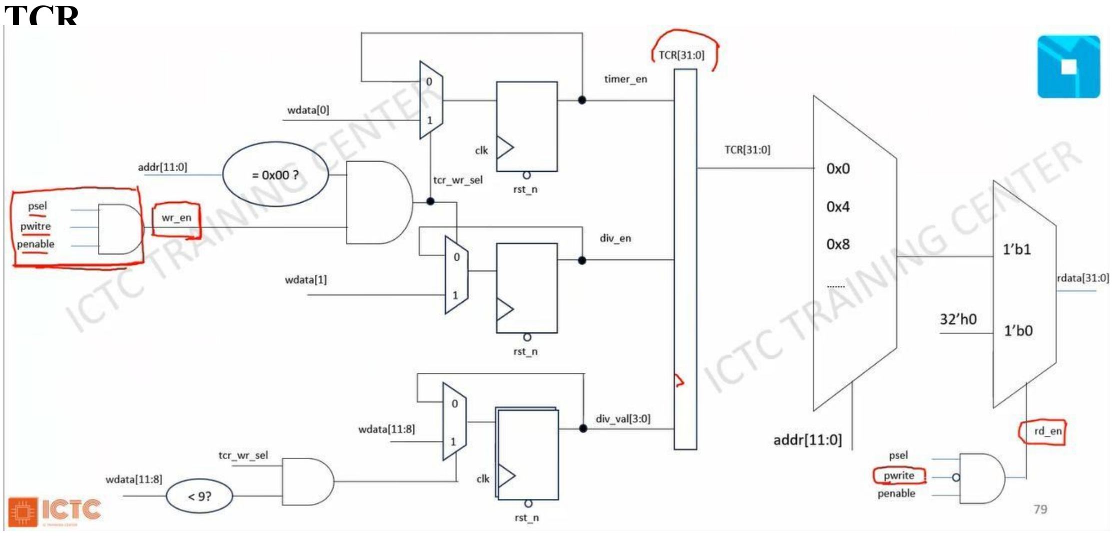
4. interrupt.
- interrupt pending (trigger condition): khi cnt = cmp
- interrupt status set: khi interrupt pending
- interrupt status clear: khi write 1 vào int_st
- interrupt output set: khi interrupt pending & interrupt is enable (int_en = 1)
- interrupt output clear: khi write 1 vào int_st hoặc disable interrupt

5. addr bit-wdith
- không phải lấy 32 bit address mà chỉ lấy số bit đủ để chứa address space thôi (cách tính như ss1)

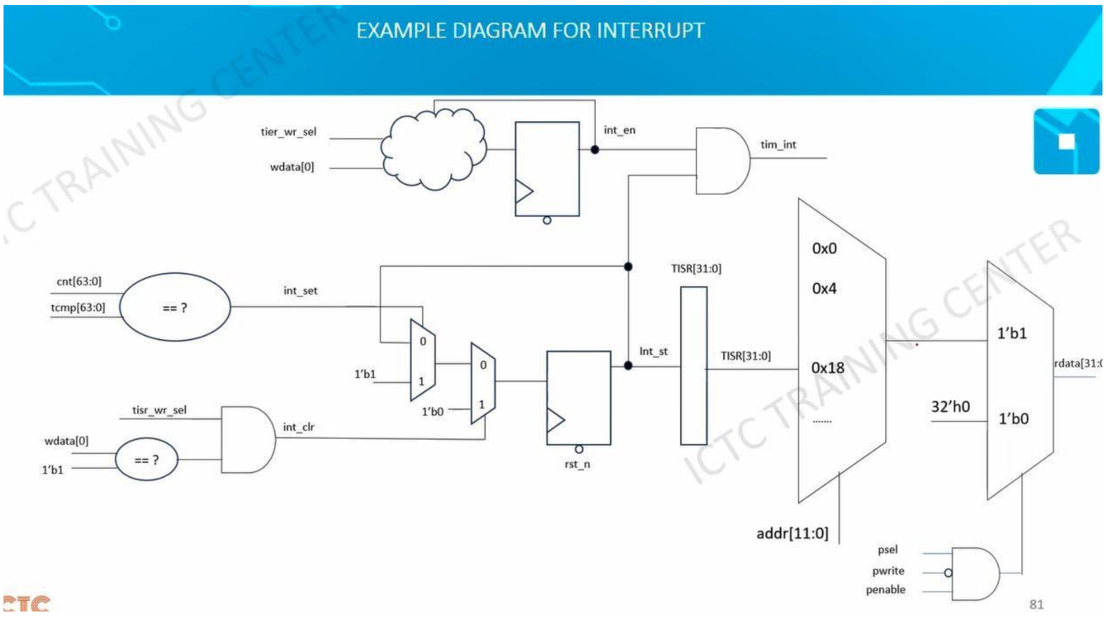
BLOCK DIAGRAMS



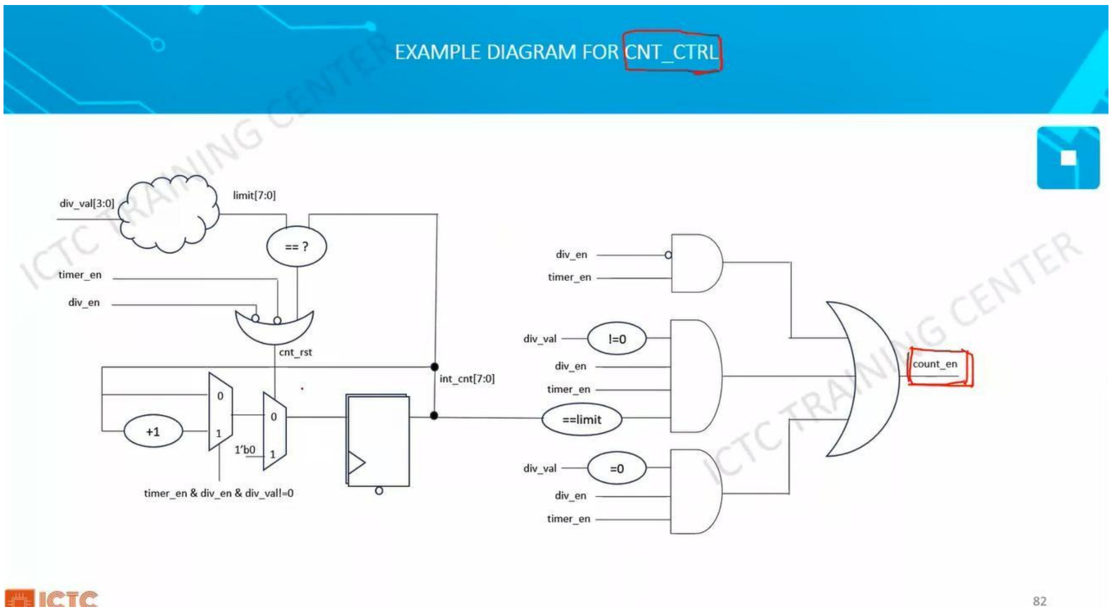
TCR



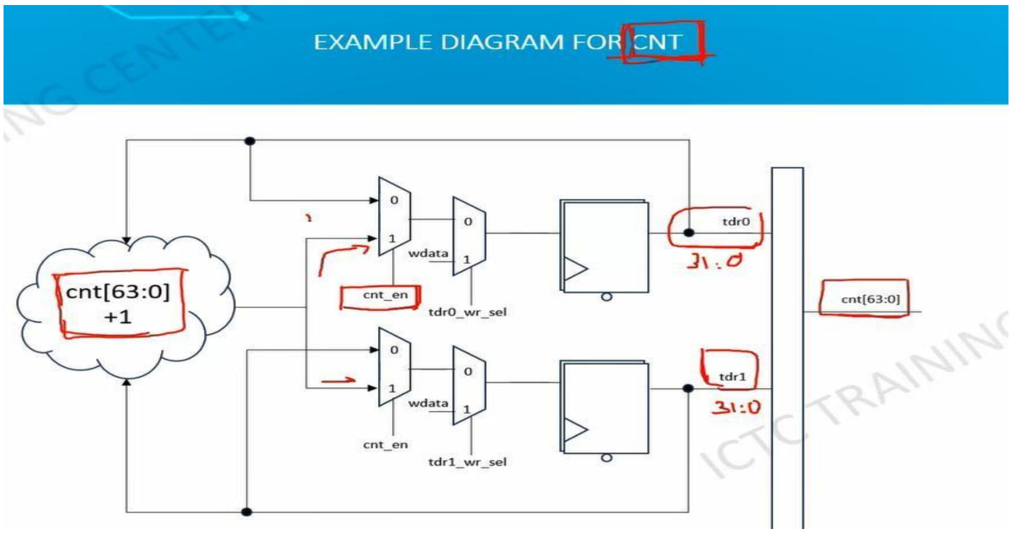
INTERRUPT



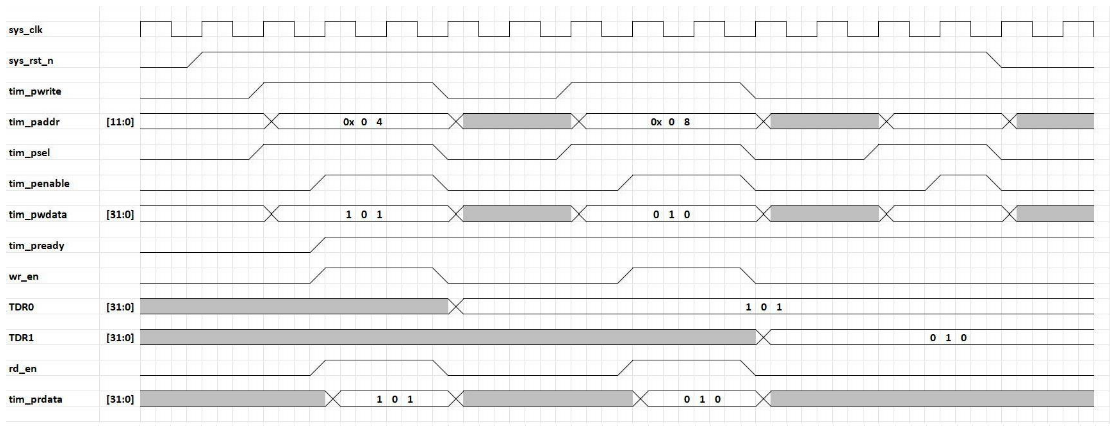
COUNT_CONTROL

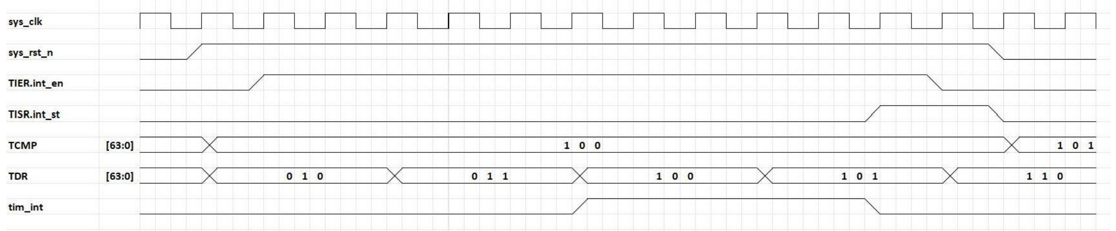


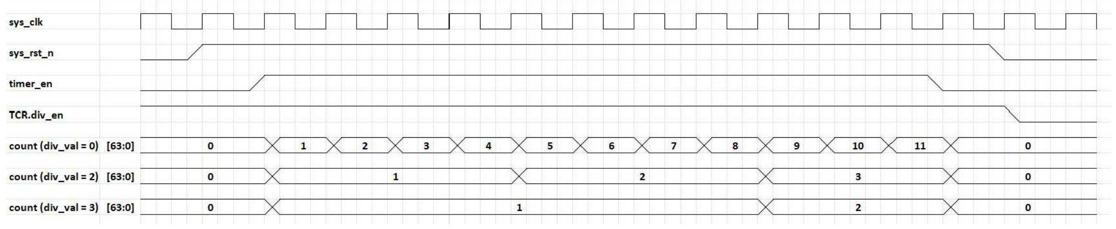
CNT



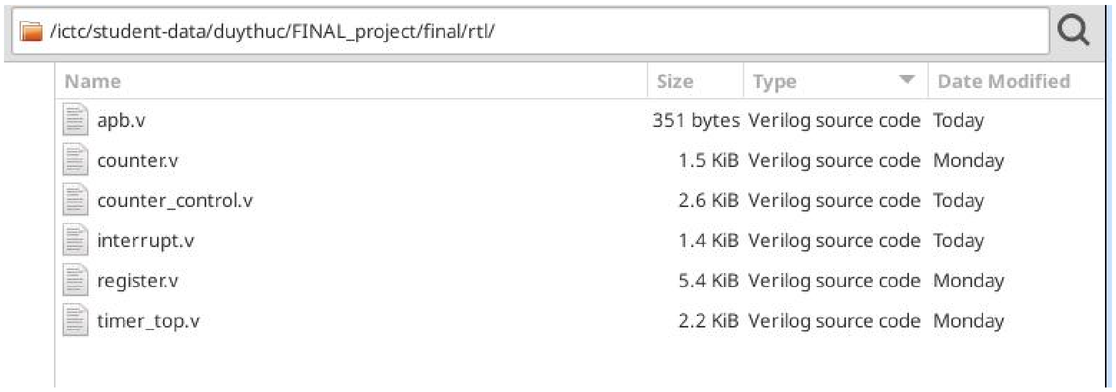
WAVEFORM diagrams






sys_clk
sys_rst_n
timer_en
TCR.div_en
count (div_val = 0) [63:0]
0 1 2 3 4 5 6 7 8 9 10 11 0
count (div_val = 2) [63:0]
0 1 2 3 0
count (div_val = 3) [63:0]
0 1 2 0

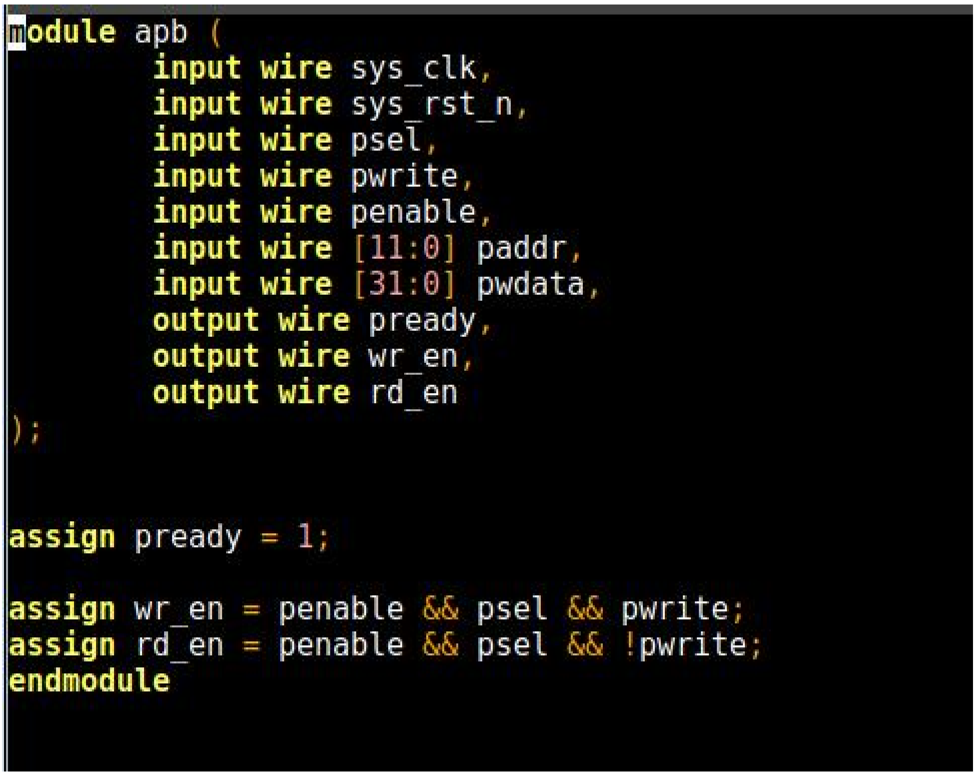
RTL code



/ictc/student-data/duythuc/FINAL_project/final/rtl/

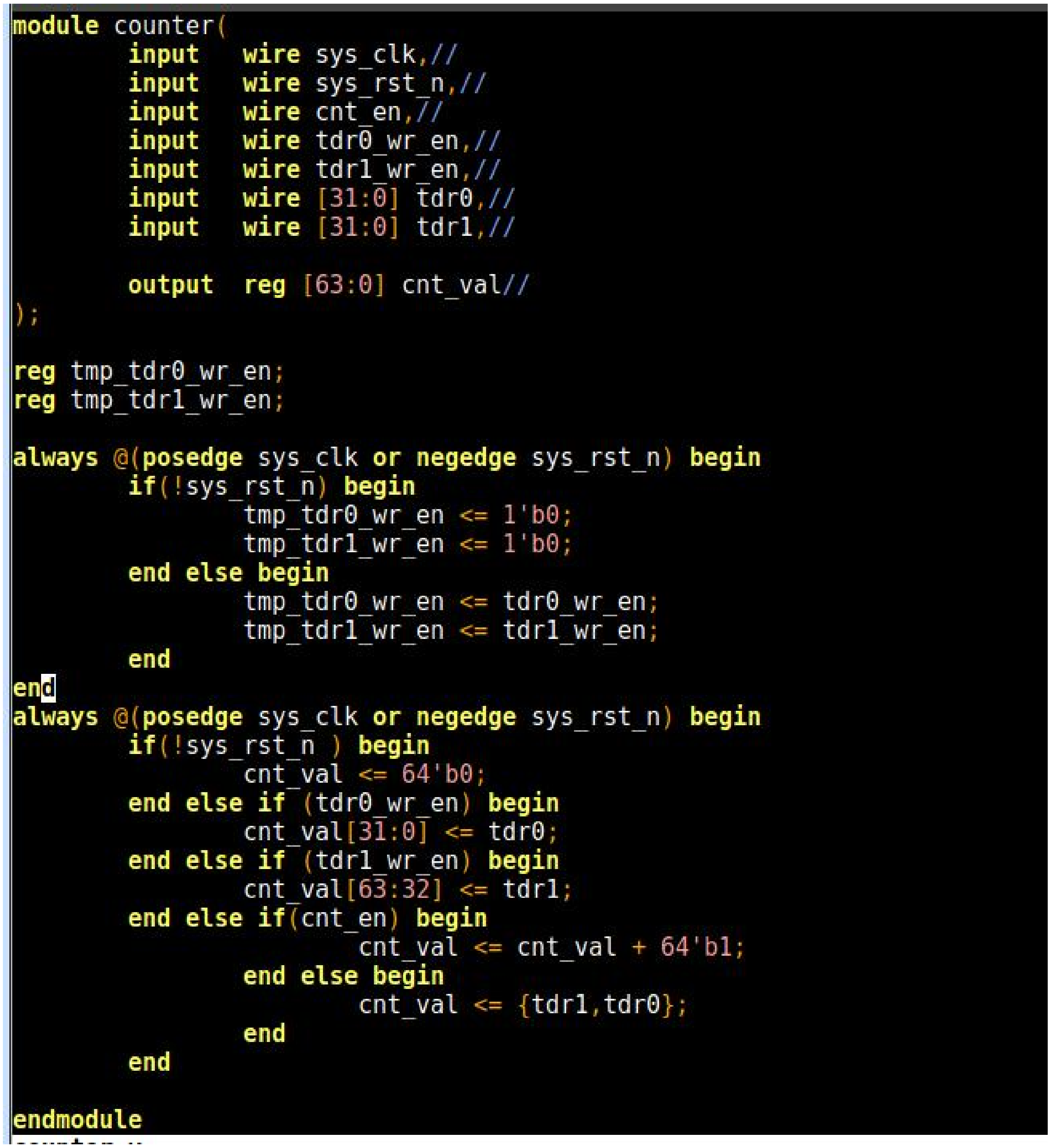
Name	Size	Type	Date Modified
apb.v	351 bytes	Verilog source code	Today
counter.v	1.5 KiB	Verilog source code	Monday
counter_control.v	2.6 KiB	Verilog source code	Today
interrupt.v	1.4 KiB	Verilog source code	Today
register.v	5.4 KiB	Verilog source code	Monday
timer_top.v	2.2 KiB	Verilog source code	Monday

Work folder with module list



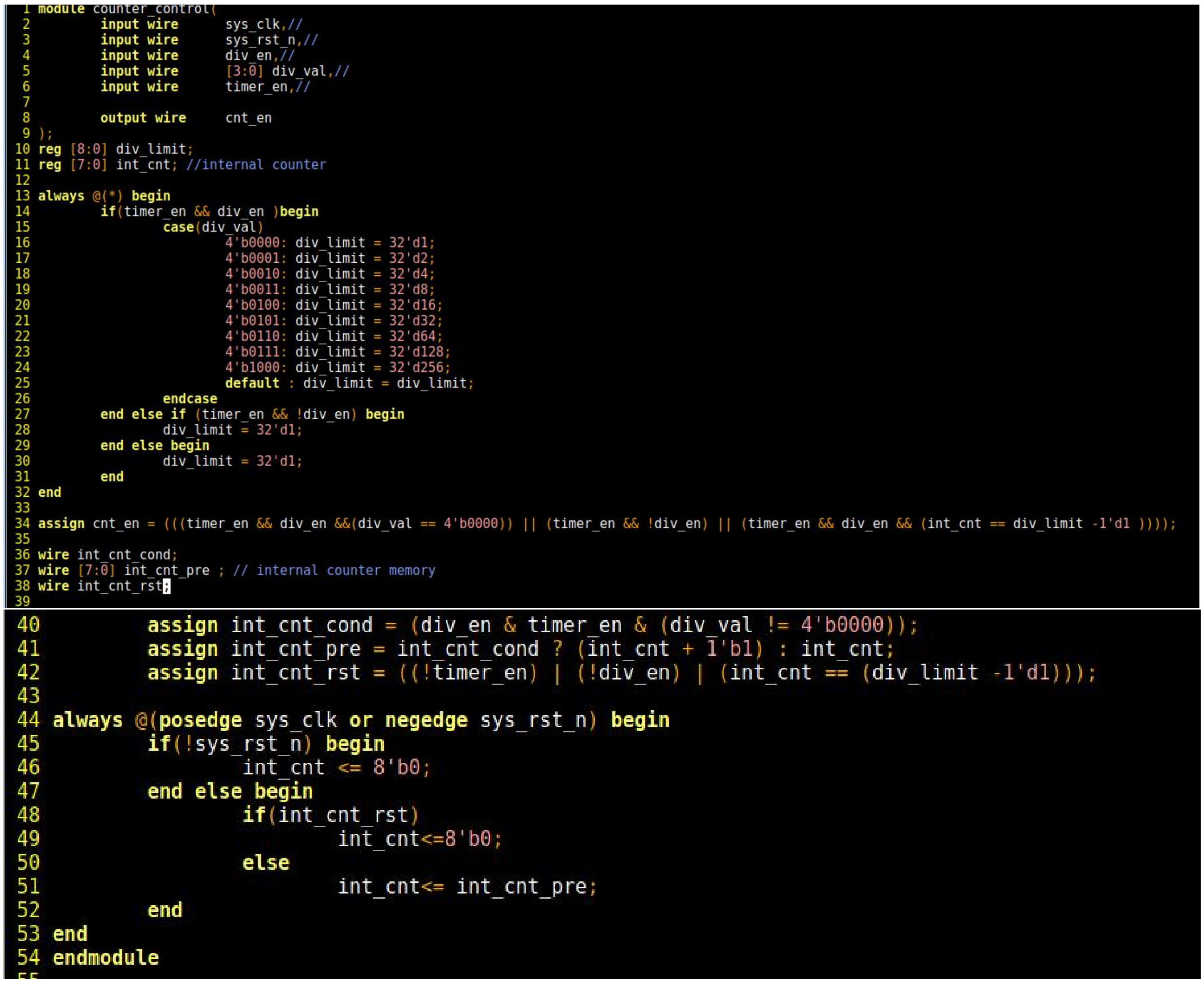
```
module apb (
        input wire sys_clk,
        input wire sys_rst_n,
        input wire psel,
        input wire pwrite,
        input wire penable,
        input wire [11:0] paddr,
        input wire [31:0] pwdata,
        output wire pready,
        output wire wr_en,
        output wire rd_en
);


assign pready = 1;

assign wr_en = penable && psel && pwrite;
assign rd_en = penable && psel && !pwrite;
endmodule
```

APB module



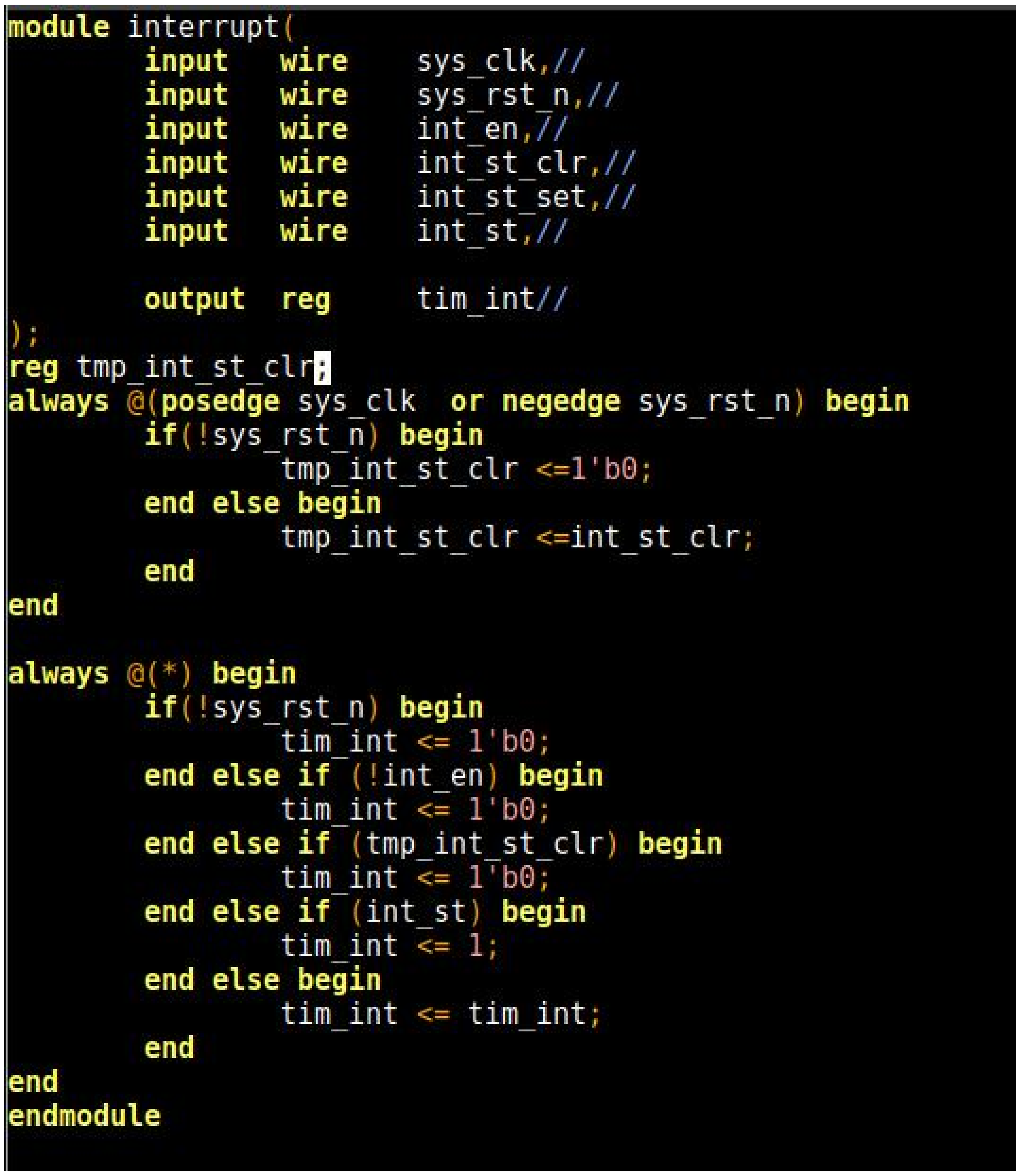
```
module counter(
        input   wire sys_clk,//
        input   wire sys_rst_n,//
        input   wire cnt_en,//
        input   wire tdr0_wr_en,//
        input   wire tdr1_wr_en,//
        input   wire [31:0] tdr0,//
        input   wire [31:0] tdr1,//

        output  reg [63:0] cnt_val//
);

reg tmp_tdr0_wr_en;
reg tmp_tdr1_wr_en;

always @(posedge sys_clk or negedge sys_rst_n) begin
        if(!sys_rst_n) begin
                tmp_tdr0_wr_en <= 1'b0;
                tmp_tdr1_wr_en <= 1'b0;
        end else begin
                tmp_tdr0_wr_en <= tdr0_wr_en;
                tmp_tdr1_wr_en <= tdr1_wr_en;
        end
end
always @(posedge sys_clk or negedge sys_rst_n) begin
        if(!sys_rst_n ) begin
                cnt_val <= 64'b0;
        end else if (tdr0_wr_en) begin
                cnt_val[31:0] <= tdr0;
        end else if (tdr1_wr_en) begin
                cnt_val[63:32] <= tdr1;
        end else if(cnt_en) begin
                        cnt_val <= cnt_val + 64'b1;
                end else begin
                        cnt_val <= {tdr1,tdr0};
                end
        end

endmodule
```

Counter module



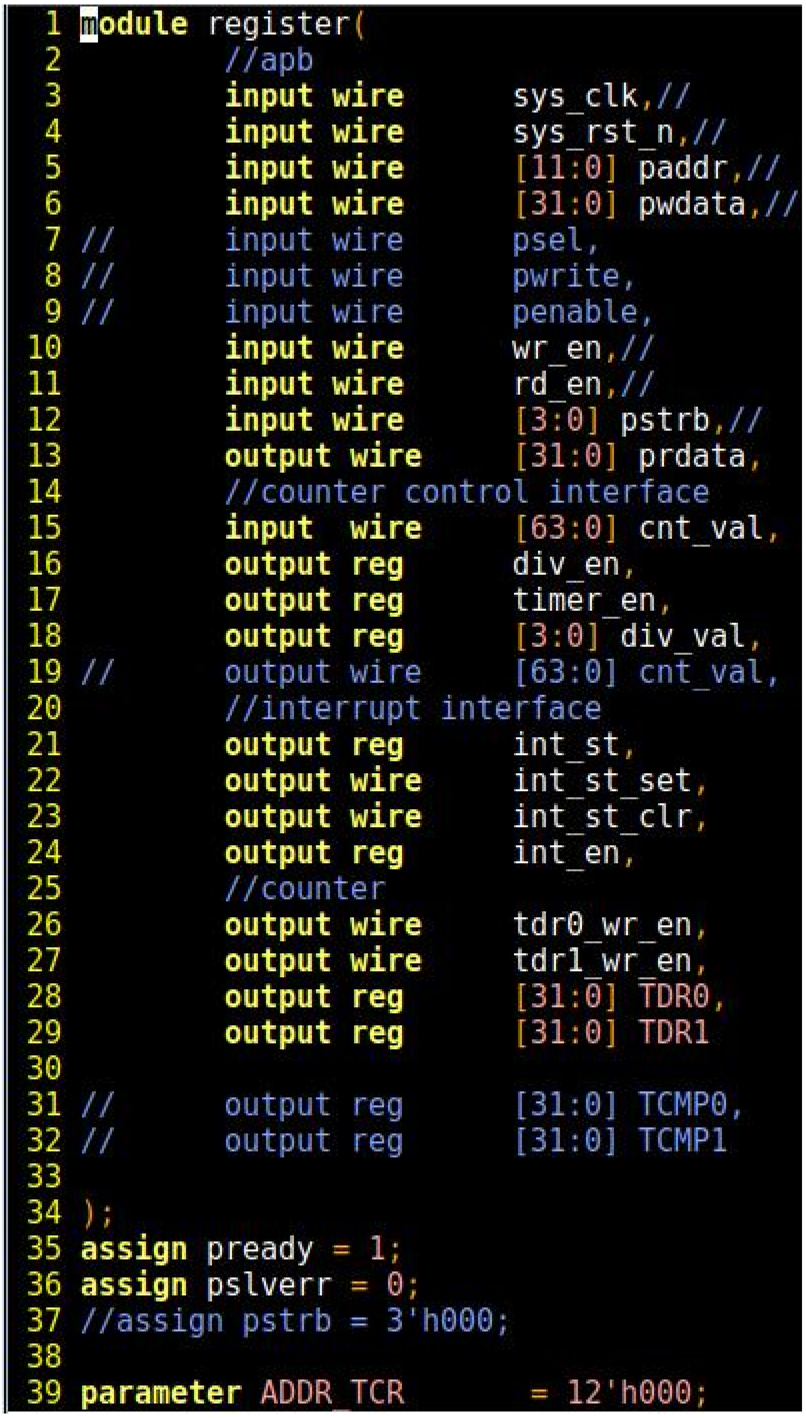
```verilog
module counter_control(
        input wire      sys_clk,//
        input wire      sys_rst_n,//
        input wire      div_en,//
        input wire      [3:0] div_val,//
        input wire      timer_en,//

        output wire     cnt_en
);
reg [8:0] div_limit;
reg [7:0] int_cnt; //internal counter

always @(*) begin
        if(timer_en && div_en )begin
                case(div_val)
                        4'b0000: div_limit = 32'd1;
                        4'b0001: div_limit = 32'd2;
                        4'b0010: div_limit = 32'd4;
                        4'b0011: div_limit = 32'd8;
                        4'b0100: div_limit = 32'd16;
                        4'b0101: div_limit = 32'd32;
                        4'b0110: div_limit = 32'd64;
                        4'b0111: div_limit = 32'd128;
                        4'b1000: div_limit = 32'd256;
                        default : div_limit = div_limit;
                endcase
        end else if (timer_en && !div_en) begin
                div_limit = 32'd1;
        end else begin
                div_limit = 32'd1;
        end
end

assign cnt_en = (((timer_en && div_en &&(div_val == 4'b0000)) || (timer_en && !div_en) || (timer_en && div_en && (int_cnt == div_limit -1'd1 ))));

wire int_cnt_cond;
wire [7:0] int_cnt_pre ; // internal counter memory
wire int_cnt_rst;

        assign int_cnt_cond = (div_en & timer_en & (div_val != 4'b0000));
        assign int_cnt_pre = int_cnt_cond ? (int_cnt + 1'b1) : int_cnt;
        assign int_cnt_rst = ((!timer_en) | (!div_en) | (int_cnt == (div_limit -1'd1)));

always @(posedge sys_clk or negedge sys_rst_n) begin
        if(!sys_rst_n) begin
                int_cnt <= 8'b0;
        end else begin
                if(int_cnt_rst)
                        int_cnt<=8'b0;
                else
                        int_cnt<= int_cnt_pre;
        end
end
endmodule
```

Counter control module



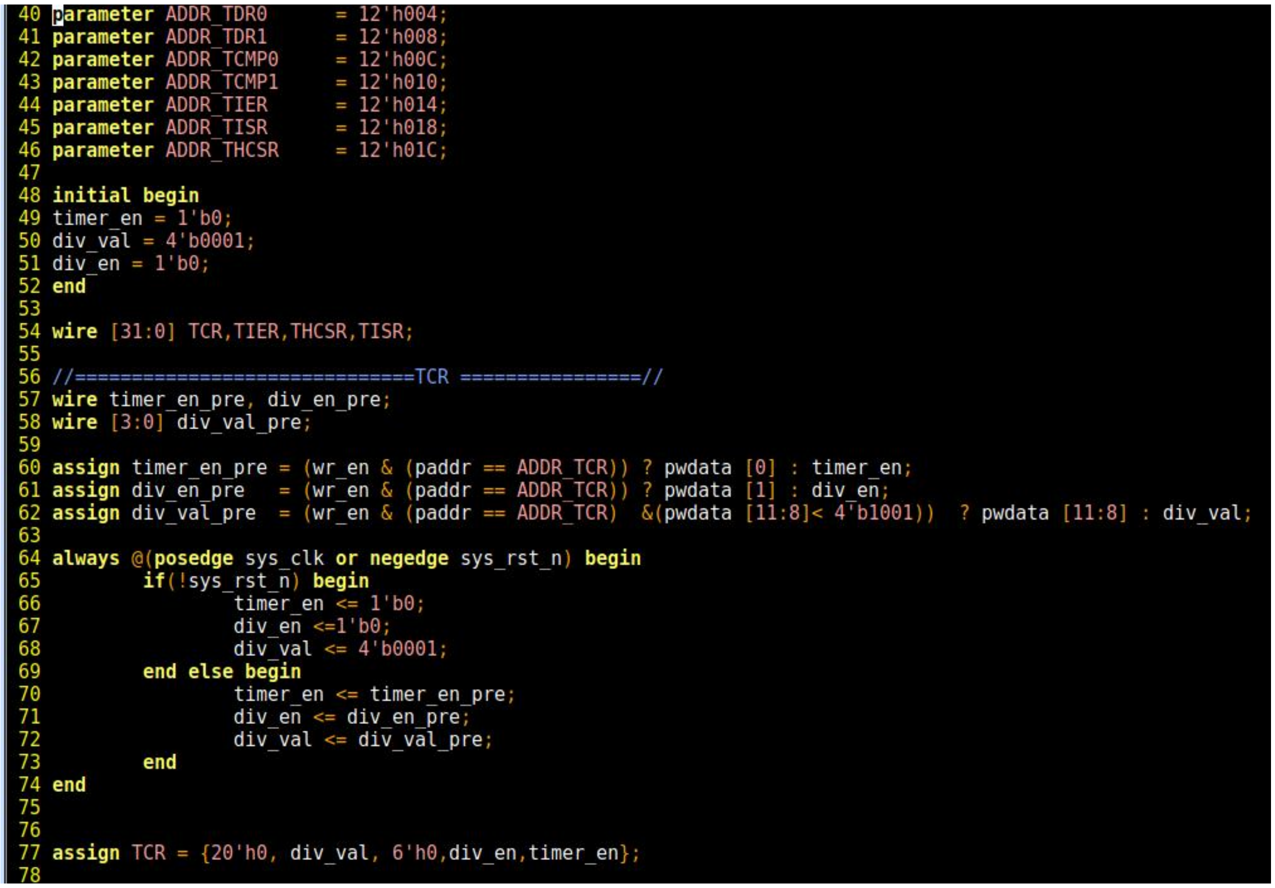
```verilog
module interrupt(
        input   wire    sys_clk,//
        input   wire    sys_rst_n,//
        input   wire    int_en,//
        input   wire    int_st_clr,//
        input   wire    int_st_set,//
        input   wire    int_st,//

        output  reg     tim_int//
);
reg tmp_int_st_clr;
always @(posedge sys_clk  or negedge sys_rst_n) begin
        if(!sys_rst_n) begin
                tmp_int_st_clr <=1'b0;
        end else begin
                tmp_int_st_clr <=int_st_clr;
        end
end

always @(*) begin
        if(!sys_rst_n) begin
                tim_int <= 1'b0;
        end else if (!int_en) begin
                tim_int <= 1'b0;
        end else if (tmp_int_st_clr) begin
                tim_int <= 1'b0;
        end else if (int_st) begin
                tim_int <= 1;
        end else begin
                tim_int <= tim_int;
        end
end
endmodule
```

Interrupt module



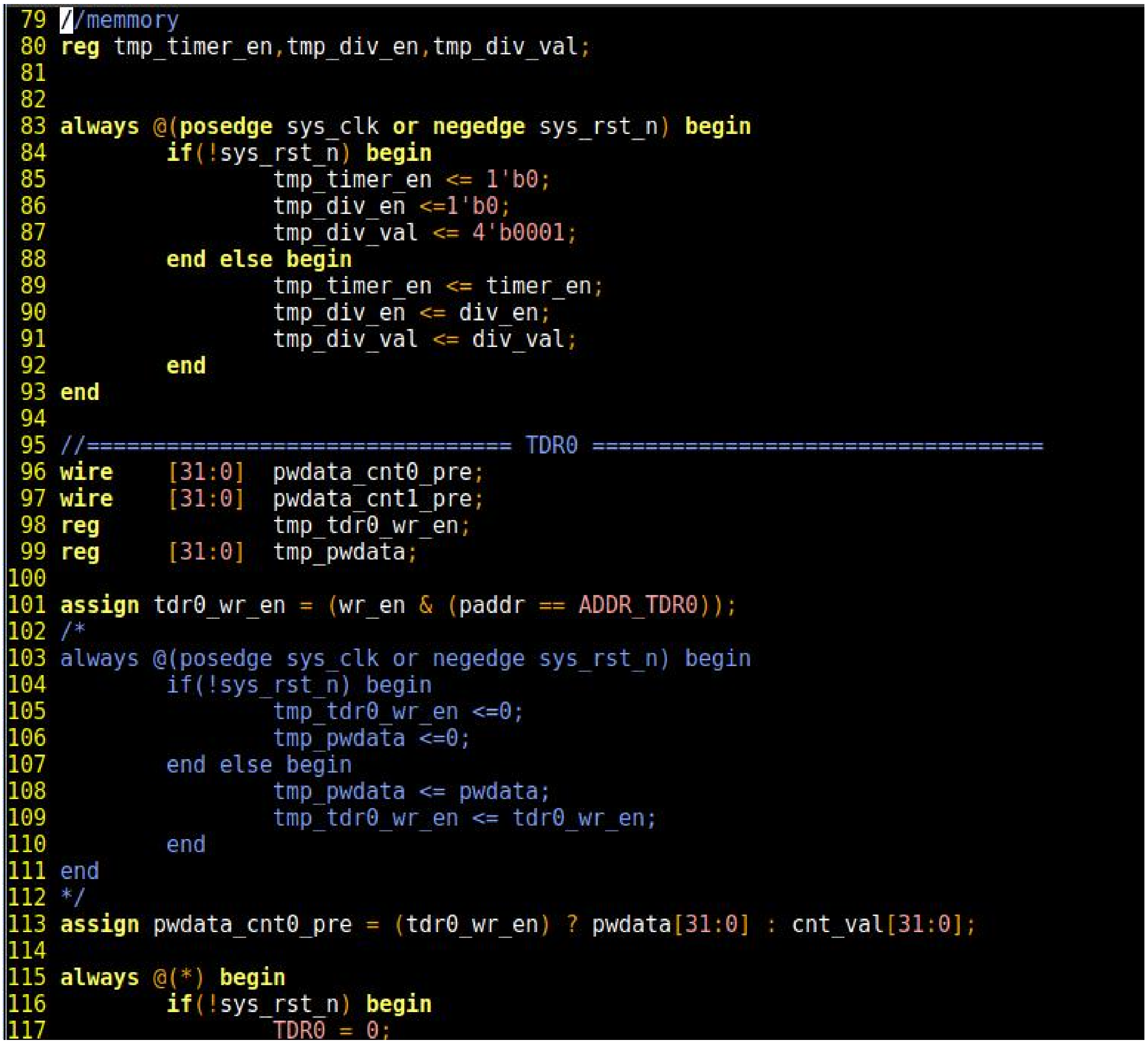
```verilog
module register(
        //apb
        input wire      sys_clk,//
        input wire      sys_rst_n,//
        input wire      [11:0] paddr,//
        input wire      [31:0] pwdata,//
//      input wire      psel,
//      input wire      pwrite,
//      input wire      penable,
        input wire      wr_en,//
        input wire      rd_en,//
        input wire      [3:0] pstrb,//
        output wire     [31:0] prdata,
        //counter control interface
        input  wire     [63:0] cnt_val,
        output reg      div_en,
        output reg      timer_en,
        output reg      [3:0] div_val,
//      output wire     [63:0] cnt_val,
        //interrupt interface
        output reg      int_st,
        output wire     int_st_set,
        output wire     int_st_clr,
        output reg      int_en,
        //counter
        output wire     tdr0_wr_en,
        output wire     tdr1_wr_en,
        output reg      [31:0] TDR0,
        output reg      [31:0] TDR1

//      output reg      [31:0] TCMP0,
//      output reg      [31:0] TCMP1

);
assign pready = 1;
assign pslverr = 0;
//assign pstrb = 3'h000;

parameter ADDR_TCR       = 12'h000;
```

Register module.1



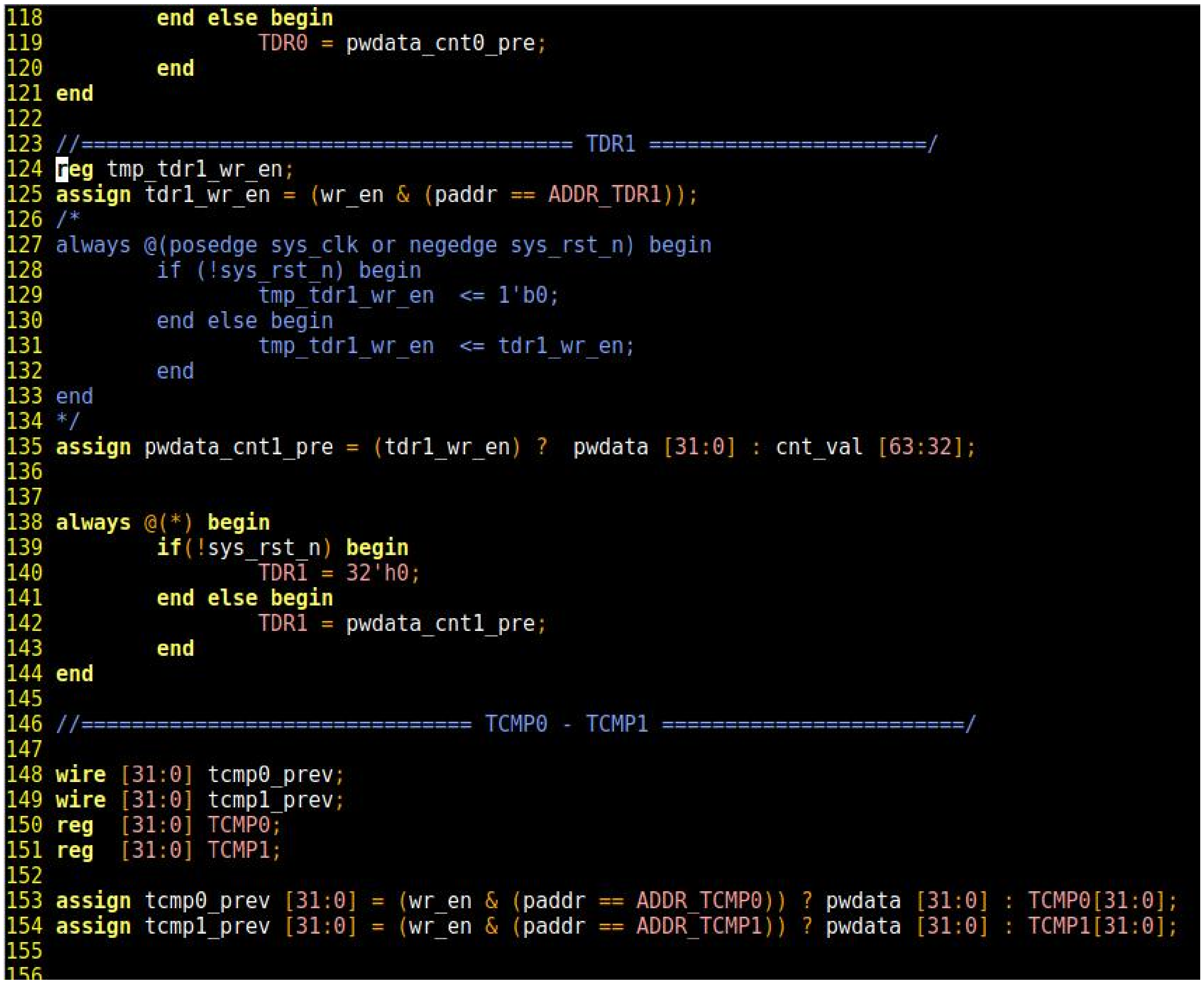
```verilog
parameter ADDR_TDR0       = 12'h004;
parameter ADDR_TDR1       = 12'h008;
parameter ADDR_TCMP0      = 12'h00C;
parameter ADDR_TCMP1      = 12'h010;
parameter ADDR_TIER       = 12'h014;
parameter ADDR_TISR       = 12'h018;
parameter ADDR_THCSR      = 12'h01C;

initial begin
timer_en = 1'b0;
div_val = 4'b0001;
div_en = 1'b0;
end

wire [31:0] TCR,TIER,THCSR,TISR;

//==============================TCR ================//
wire timer_en_pre, div_en_pre;
wire [3:0] div_val_pre;

assign timer_en_pre = (wr_en & (paddr == ADDR_TCR)) ? pwdata [0] : timer_en;
assign div_en_pre   = (wr_en & (paddr == ADDR_TCR)) ? pwdata [1] : div_en;
assign div_val_pre  = (wr_en & (paddr == ADDR_TCR)  &(pwdata [11:8]< 4'b1001))  ? pwdata [11:8] : div_val;

always @(posedge sys_clk or negedge sys_rst_n) begin
        if(!sys_rst_n) begin
                timer_en <= 1'b0;
                div_en <=1'b0;
                div_val <= 4'b0001;
        end else begin
                timer_en <= timer_en_pre;
                div_en <= div_en_pre;
                div_val <= div_val_pre;
        end
end


assign TCR = {20'h0, div_val, 6'h0,div_en,timer_en};

```

Register module.2



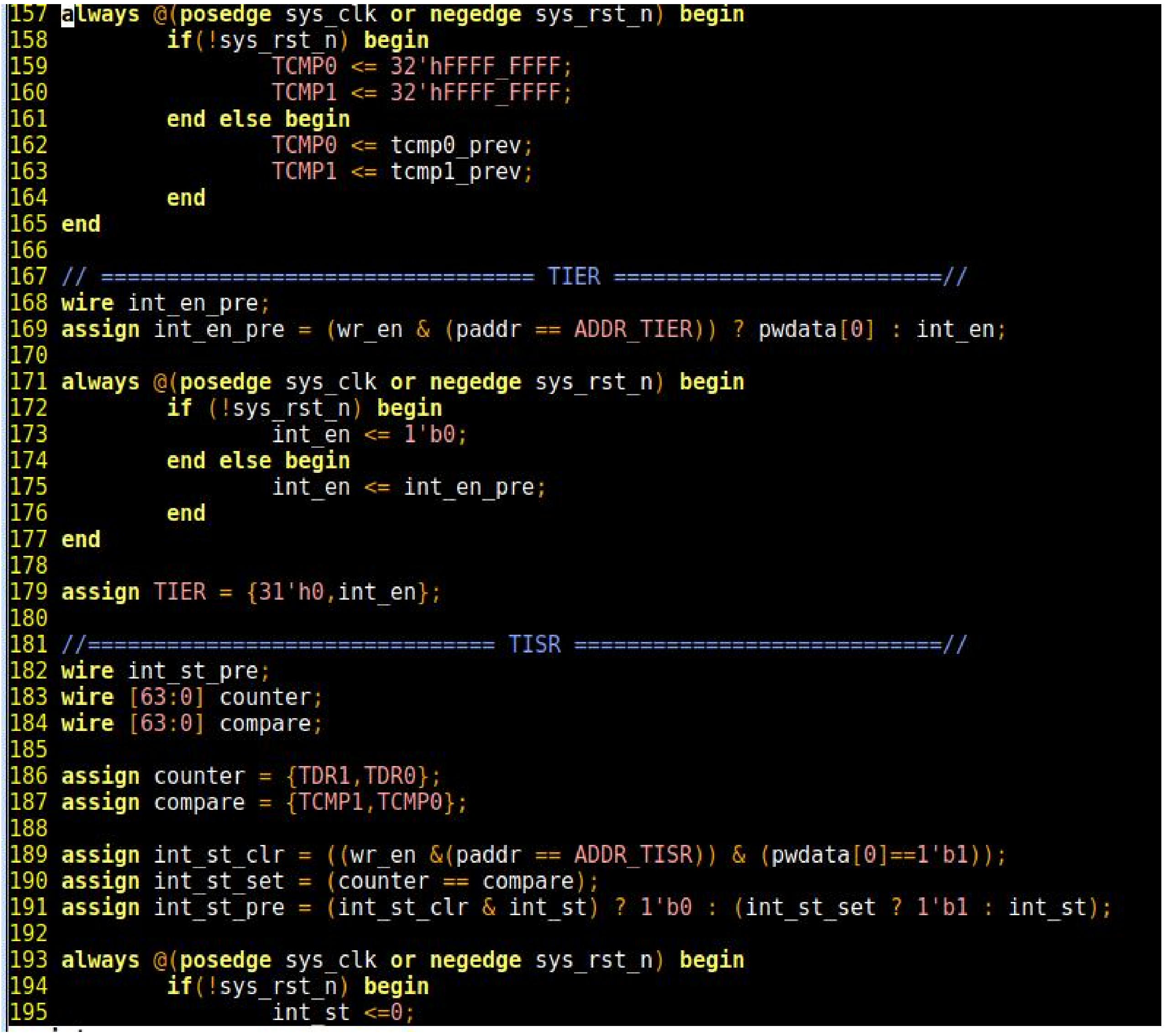
```
//memmory
reg tmp_timer_en,tmp_div_en,tmp_div_val;


always @(posedge sys_clk or negedge sys_rst_n) begin
        if(!sys_rst_n) begin
                tmp_timer_en <= 1'b0;
                tmp_div_en <=1'b0;
                tmp_div_val <= 4'b0001;
        end else begin
                tmp_timer_en <= timer_en;
                tmp_div_en <= div_en;
                tmp_div_val <= div_val;
        end
end

//================================ TDR0 ==================================
wire    [31:0]  pwdata_cnt0_pre;
wire    [31:0]  pwdata_cnt1_pre;
reg             tmp_tdr0_wr_en;
reg     [31:0]  tmp_pwdata;

assign tdr0_wr_en = (wr_en & (paddr == ADDR_TDR0));
/*
always @(posedge sys_clk or negedge sys_rst_n) begin
        if(!sys_rst_n) begin
                tmp_tdr0_wr_en <=0;
                tmp_pwdata <=0;
        end else begin
                tmp_pwdata <= pwdata;
                tmp_tdr0_wr_en <= tdr0_wr_en;
        end
end
*/
assign pwdata_cnt0_pre = (tdr0_wr_en) ? pwdata[31:0] : cnt_val[31:0];

always @(*) begin
        if(!sys_rst_n) begin
                TDR0 = 0;
```

Register module.3



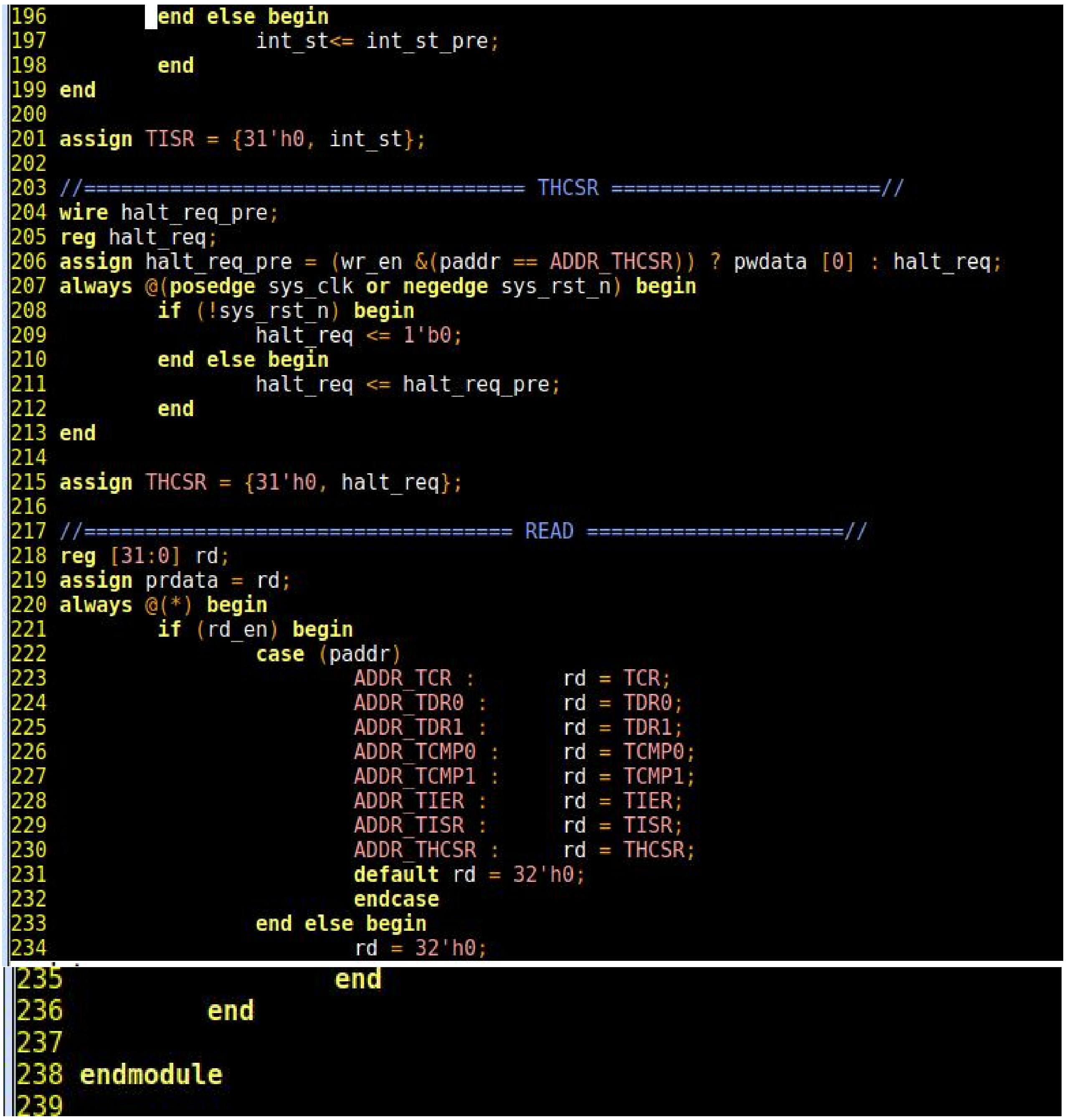
```
        end else begin
                TDR0 = pwdata_cnt0_pre;
        end
end

//======================================== TDR1 ======================/
reg tmp_tdr1_wr_en;
assign tdr1_wr_en = (wr_en & (paddr == ADDR_TDR1));
/*
always @(posedge sys_clk or negedge sys_rst_n) begin
        if (!sys_rst_n) begin
                tmp_tdr1_wr_en  <= 1'b0;
        end else begin
                tmp_tdr1_wr_en  <= tdr1_wr_en;
        end
end
*/
assign pwdata_cnt1_pre = (tdr1_wr_en) ?  pwdata [31:0] : cnt_val [63:32];


always @(*) begin
        if(!sys_rst_n) begin
                TDR1 = 32'h0;
        end else begin
                TDR1 = pwdata_cnt1_pre;
        end
end

//=============================== TCMP0 - TCMP1 =======================/

wire [31:0] tcmp0_prev;
wire [31:0] tcmp1_prev;
reg  [31:0] TCMP0;
reg  [31:0] TCMP1;

assign tcmp0_prev [31:0] = (wr_en & (paddr == ADDR_TCMP0)) ? pwdata [31:0] : TCMP0[31:0];
assign tcmp1_prev [31:0] = (wr_en & (paddr == ADDR_TCMP1)) ? pwdata [31:0] : TCMP1[31:0];

```

Register module.4



```
always @(posedge sys_clk or negedge sys_rst_n) begin
        if(!sys_rst_n) begin
                TCMP0 <= 32'hFFFF_FFFF;
                TCMP1 <= 32'hFFFF_FFFF;
        end else begin
                TCMP0 <= tcmp0_prev;
                TCMP1 <= tcmp1_prev;
        end
end

// ================================= TIER =========================//
wire int_en_pre;
assign int_en_pre = (wr_en & (paddr == ADDR_TIER)) ? pwdata[0] : int_en;

always @(posedge sys_clk or negedge sys_rst_n) begin
        if (!sys_rst_n) begin
                int_en <= 1'b0;
        end else begin
                int_en <= int_en_pre;
        end
end

assign TIER = {31'h0,int_en};

//================================ TISR ============================//
wire int_st_pre;
wire [63:0] counter;
wire [63:0] compare;

assign counter = {TDR1,TDR0};
assign compare = {TCMP1,TCMP0};

assign int_st_clr = ((wr_en &(paddr == ADDR_TISR)) & (pwdata[0]==1'b1));
assign int_st_set = (counter == compare);
assign int_st_pre = (int_st_clr & int_st) ? 1'b0 : (int_st_set ? 1'b1 : int_st);

always @(posedge sys_clk or negedge sys_rst_n) begin
        if(!sys_rst_n) begin
                int_st <=0;
```

Register module.5



```
        end else begin
                int_st<= int_st_pre;
        end
end

assign TISR = {31'h0, int_st};

//==================================== THCSR ======================//
wire halt_req_pre;
reg halt_req;
assign halt_req_pre = (wr_en &(paddr == ADDR_THCSR)) ? pwdata [0] : halt_req;
always @(posedge sys_clk or negedge sys_rst_n) begin
        if (!sys_rst_n) begin
                halt_req <= 1'b0;
        end else begin
                halt_req <= halt_req_pre;
        end
end

assign THCSR = {31'h0, halt_req};

//=================================== READ =====================//
reg [31:0] rd;
assign prdata = rd;
always @(*) begin
        if (rd_en) begin
                case (paddr)
                        ADDR_TCR :      rd = TCR;
                        ADDR_TDR0 :     rd = TDR0;
                        ADDR_TDR1 :     rd = TDR1;
                        ADDR_TCMP0 :    rd = TCMP0;
                        ADDR_TCMP1 :    rd = TCMP1;
                        ADDR_TIER :     rd = TIER;
                        ADDR_TISR :     rd = TISR;
                        ADDR_THCSR :    rd = THCSR;
                        default rd = 32'h0;
                        endcase
                end else begin
                        rd = 32'h0;
                end
        end

endmodule

```

Register module.6

VERIFICATION

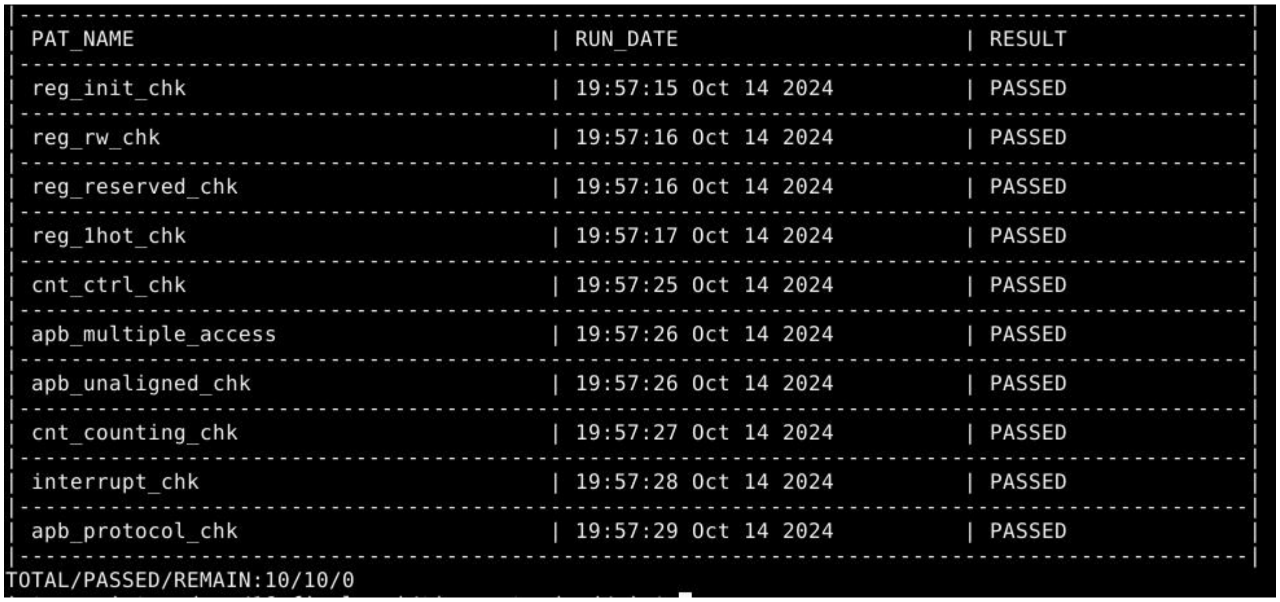
CHECK LIST :

https://docs.google.com/spreadsheets/d/1eWd2OWbPPiDY-ZPL8ft7e0dhWcnSQxPl/edit?usp=drive_link&ouid=100313224507677416115&rtpof=true&sd=true

OVERALL POINT :

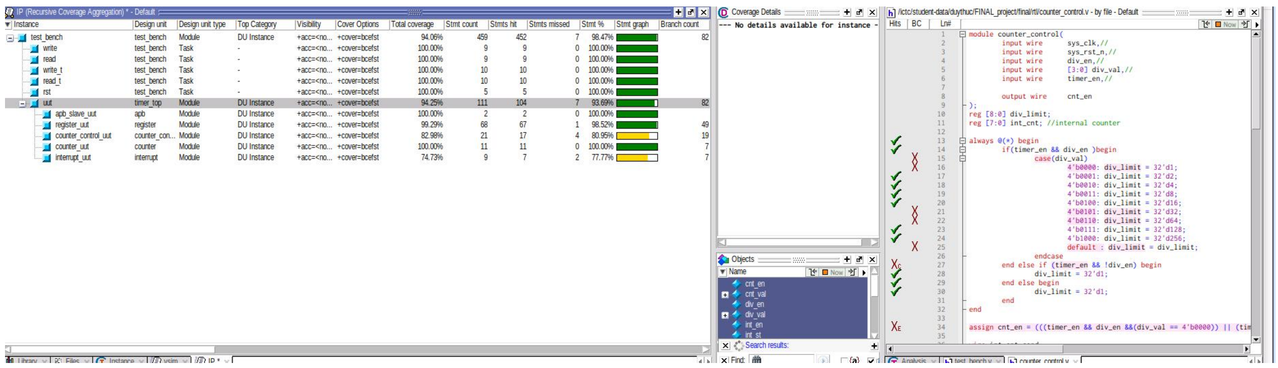
https://docs.google.com/spreadsheets/d/1w1LRGGeeL9DDijFJcNTZcvBHxJluU6Bs6mi8y-xhDU8/edit?usp=drive_link

RESULT :



```
|-------------------------------------------------------------------------------------------------|
| PAT_NAME                                  | RUN_DATE                       | RESULT              |
|-------------------------------------------------------------------------------------------------|
| reg_init_chk                              | 19:57:15 Oct 14 2024           | PASSED              |
|-------------------------------------------------------------------------------------------------|
| reg_rw_chk                                | 19:57:16 Oct 14 2024           | PASSED              |
|-------------------------------------------------------------------------------------------------|
| reg_reserved_chk                          | 19:57:16 Oct 14 2024           | PASSED              |
|-------------------------------------------------------------------------------------------------|
| reg_1hot_chk                              | 19:57:17 Oct 14 2024           | PASSED              |
|-------------------------------------------------------------------------------------------------|
| cnt_ctrl_chk                              | 19:57:25 Oct 14 2024           | PASSED              |
|-------------------------------------------------------------------------------------------------|
| apb_multiple_access                       | 19:57:26 Oct 14 2024           | PASSED              |
|-------------------------------------------------------------------------------------------------|
| apb_unaligned_chk                         | 19:57:26 Oct 14 2024           | PASSED              |
|-------------------------------------------------------------------------------------------------|
| cnt_counting_chk                          | 19:57:27 Oct 14 2024           | PASSED              |
|-------------------------------------------------------------------------------------------------|
| interrupt_chk                             | 19:57:28 Oct 14 2024           | PASSED              |
|-------------------------------------------------------------------------------------------------|
| apb_protocol_chk                          | 19:57:29 Oct 14 2024           | PASSED              |
|-------------------------------------------------------------------------------------------------|
TOTAL/PASSED/REMAIN:10/10/0
```

Golden model 10/10 PASSED



IP (Recursive Coverage Aggregation) * - Default

Instance	Design unit	Design unit type	Top Category	Visibility	Cover Options	Total coverage	Stmt count	Stmts hit	Stmts missed	Stmt %	Stmt graph	Branch count
test_bench	test_bench	Module	DU Instance	+acc=<no...	+cover=bcefst	94.06%	459	452	7	98.47%	🟩	82
write	test_bench	Task	-	+acc=<no...	+cover=bcefst	100.00%	9	9	0	100.00%	🟩	
read	test_bench	Task	-	+acc=<no...	+cover=bcefst	100.00%	9	9	0	100.00%	🟩	
write_t	test_bench	Task	-	+acc=<no...	+cover=bcefst	100.00%	10	10	0	100.00%	🟩	
read_t	test_bench	Task	-	+acc=<no...	+cover=bcefst	100.00%	10	10	0	100.00%	🟩	
rst	test_bench	Task	-	+acc=<no...	+cover=bcefst	100.00%	5	5	0	100.00%	🟩	
uut	timer_top	Module	DU Instance	+acc=<no...	+cover=bcefst	94.25%	111	104	7	93.69%	🟩	82
apb_slave_uut	apb	Module	DU Instance	+acc=<no...	+cover=bcefst	100.00%	2	2	0	100.00%	🟩	
register_uut	register	Module	DU Instance	+acc=<no...	+cover=bcefst	99.29%	68	67	1	98.52%	🟩	49
counter_control_uut	counter_con...	Module	DU Instance	+acc=<no...	+cover=bcefst	82.98%	21	17	4	80.95%	🟨	19
counter_uut	counter	Module	DU Instance	+acc=<no...	+cover=bcefst	100.00%	11	11	0	100.00%	🟩	7
interrupt_uut	interrupt	Module	DU Instance	+acc=<no...	+cover=bcefst	74.73%	9	7	2	77.77%	🟨	7

Coverage Details

--- No details available for instance -

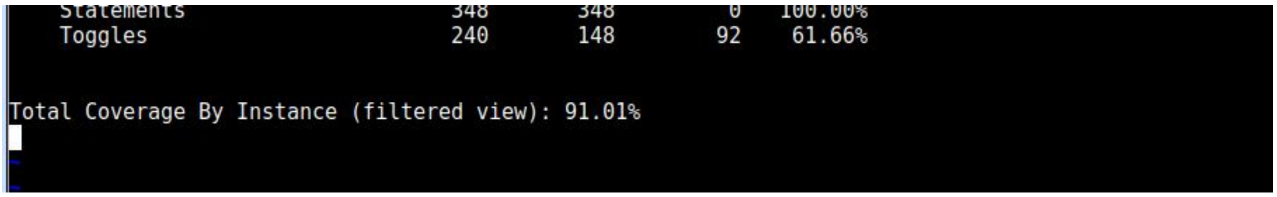
Objects

cnt_en
cnt_val
div_en
div_val
int_en
int_st

/ictc/student-data/duythuc/FINAL_project/final/rtl/counter_control.v - by file - Default

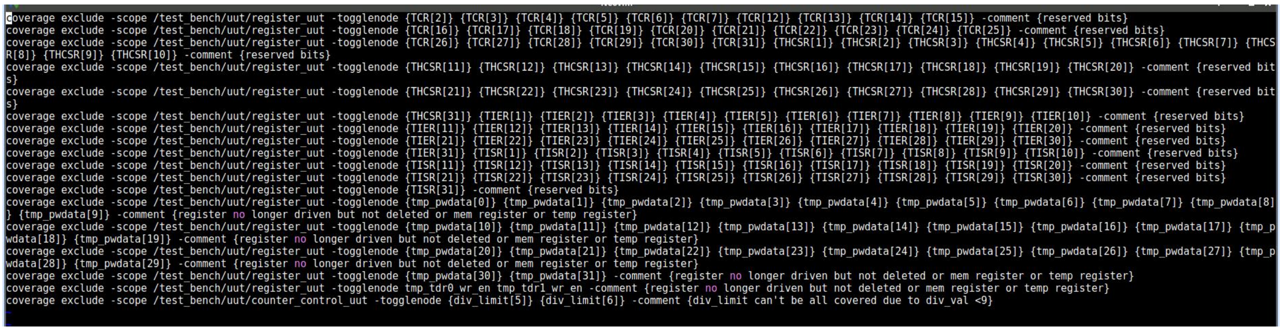
```
module counter_control(
        input wire      sys_clk,//
        input wire      sys_rst_n,//
        input wire      div_en,//
        input wire      [3:0] div_val,//
        input wire      timer_en,//

        output wire     cnt_en
);
reg [8:0] div_limit;
reg [7:0] int_cnt; //internal counter

always @(*) begin
        if(timer_en && div_en )begin
                case(div_val)
                        4'b0000: div_limit = 32'd1;
                        4'b0001: div_limit = 32'd2;
                        4'b0010: div_limit = 32'd4;
                        4'b0011: div_limit = 32'd8;
                        4'b0100: div_limit = 32'd16;
                        4'b0101: div_limit = 32'd32;
                        4'b0110: div_limit = 32'd64;
                        4'b0111: div_limit = 32'd128;
                        4'b1000: div_limit = 32'd256;
                        default : div_limit = div_limit;
                endcase
        end else if (timer_en && !div_en) begin
                div_limit = 32'd1;
        end else begin
                div_limit = 32'd1;
        end
end

assign cnt_en = (((timer_en && div_en &&(div_val == 4'b0000)) || (tim
```

98% total coverage



```
    Statements                      348        348         0    100.00%
    Toggles                         240        148        92     61.66%


Total Coverage By Instance (filtered view): 91.01%
```

91% summary coverage report



```
coverage exclude -scope /test_bench/uut/register_uut -togglenode {TCR[2]} {TCR[3]} {TCR[4]} {TCR[5]} {TCR[6]} {TCR[7]} {TCR[12]} {TCR[13]} {TCR[14]} {TCR[15]} -comment {reserved bits}
coverage exclude -scope /test_bench/uut/register_uut -togglenode {TCR[16]} {TCR[17]} {TCR[18]} {TCR[19]} {TCR[20]} {TCR[21]} {TCR[22]} {TCR[23]} {TCR[24]} {TCR[25]} -comment {reserved bits}
coverage exclude -scope /test_bench/uut/register_uut -togglenode {TCR[26]} {TCR[27]} {TCR[28]} {TCR[29]} {TCR[30]} {TCR[31]} {THCSR[1]} {THCSR[2]} {THCSR[3]} {THCSR[4]} {THCSR[5]} {THCSR[6]} {THCSR[7]} {THCS
R[8]} {THCSR[9]} {THCSR[10]} -comment {reserved bits}
coverage exclude -scope /test_bench/uut/register_uut -togglenode {THCSR[11]} {THCSR[12]} {THCSR[13]} {THCSR[14]} {THCSR[15]} {THCSR[16]} {THCSR[17]} {THCSR[18]} {THCSR[19]} {THCSR[20]} -comment {reserved bit
s}
coverage exclude -scope /test_bench/uut/register_uut -togglenode {THCSR[21]} {THCSR[22]} {THCSR[23]} {THCSR[24]} {THCSR[25]} {THCSR[26]} {THCSR[27]} {THCSR[28]} {THCSR[29]} {THCSR[30]} -comment {reserved bit
s}
coverage exclude -scope /test_bench/uut/register_uut -togglenode {THCSR[31]} {TIER[1]} {TIER[2]} {TIER[3]} {TIER[4]} {TIER[5]} {TIER[6]} {TIER[7]} {TIER[8]} {TIER[9]} {TIER[10]} -comment {reserved bits}
coverage exclude -scope /test_bench/uut/register_uut -togglenode {TIER[11]} {TIER[12]} {TIER[13]} {TIER[14]} {TIER[15]} {TIER[16]} {TIER[17]} {TIER[18]} {TIER[19]} {TIER[20]} -comment {reserved bits}
coverage exclude -scope /test_bench/uut/register_uut -togglenode {TIER[21]} {TIER[22]} {TIER[23]} {TIER[24]} {TIER[25]} {TIER[26]} {TIER[27]} {TIER[28]} {TIER[29]} {TIER[30]} -comment {reserved bits}
coverage exclude -scope /test_bench/uut/register_uut -togglenode {TIER[31]} {TISR[1]} {TISR[2]} {TISR[3]} {TISR[4]} {TISR[5]} {TISR[6]} {TISR[7]} {TISR[8]} {TISR[9]} {TISR[10]} -comment {reserved bits}
coverage exclude -scope /test_bench/uut/register_uut -togglenode {TISR[11]} {TISR[12]} {TISR[13]} {TISR[14]} {TISR[15]} {TISR[16]} {TISR[17]} {TISR[18]} {TISR[19]} {TISR[20]} -comment {reserved bits}
coverage exclude -scope /test_bench/uut/register_uut -togglenode {TISR[21]} {TISR[22]} {TISR[23]} {TISR[24]} {TISR[25]} {TISR[26]} {TISR[27]} {TISR[28]} {TISR[29]} {TISR[30]} -comment {reserved bits}
coverage exclude -scope /test_bench/uut/register_uut -togglenode {TISR[31]} -comment {reserved bits}
coverage exclude -scope /test_bench/uut/register_uut -togglenode {tmp_pwdata[0]} {tmp_pwdata[1]} {tmp_pwdata[2]} {tmp_pwdata[3]} {tmp_pwdata[4]} {tmp_pwdata[5]} {tmp_pwdata[6]} {tmp_pwdata[7]} {tmp_pwdata[8]
} {tmp_pwdata[9]} -comment {register no longer driven but not deleted or mem register or temp register}
coverage exclude -scope /test_bench/uut/register_uut -togglenode {tmp_pwdata[10]} {tmp_pwdata[11]} {tmp_pwdata[12]} {tmp_pwdata[13]} {tmp_pwdata[14]} {tmp_pwdata[15]} {tmp_pwdata[16]} {tmp_pwdata[17]} {tmp_p
wdata[18]} {tmp_pwdata[19]} -comment {register no longer driven but not deleted or mem register or temp register}
coverage exclude -scope /test_bench/uut/register_uut -togglenode {tmp_pwdata[20]} {tmp_pwdata[21]} {tmp_pwdata[22]} {tmp_pwdata[23]} {tmp_pwdata[24]} {tmp_pwdata[25]} {tmp_pwdata[26]} {tmp_pwdata[27]} {tmp_p
wdata[28]} {tmp_pwdata[29]} -comment {register no longer driven but not deleted or mem register or temp register}
coverage exclude -scope /test_bench/uut/register_uut -togglenode {tmp_pwdata[30]} {tmp_pwdata[31]} -comment {register no longer driven but not deleted or mem register or temp register}
coverage exclude -scope /test_bench/uut/register_uut -togglenode tmp_tdr0_wr_en tmp_tdr1_wr_en -comment {register no longer driven but not deleted or mem register or temp register}
coverage exclude -scope /test_bench/uut/counter_control_uut -togglenode {div_limit[5]} {div_limit[6]} -comment {div_limit can't be all covered due to div_val <9}
```

Exclude.doc